

# A Survey on Algorithms for Euclidean Matching

SAYAN BHATTACHARYA\*  
Computational Geometry Project

## 1 Introduction

Euclidean minimum cost perfect matching is a well studied problem in Computational Geometry, having manifold applications in VLSI, statistics, pattern recognition and operations research. We mention several results and summarize some of the algorithms proposed for this problem over the course of last two decades.

### 1.1 Problem Formulation

Let  $G = (V, E)$  be a weighted undirected graph with vertex set  $V$ , edge set  $E$  and a weight function  $d$ . Thus,  $d(u, v)$  denotes the weight of any edge  $(u, v) \in E$ . A *matching*  $M \subseteq E$  is a collection of edges such that every node in  $V$  is incident to at most one edge in  $M$ . The matching is *perfect* if every node in  $V$  is incident to exactly one edge in  $M$ . The *cost* of the matching is given by  $\sum_{(u,v) \in M} d(u, v)$ .

The Euclidean Bipartite Matching Problem (EBM) is defined as follows: Given a set  $R$  of  $n$  *red* points and another set  $B$  of  $n$  *blue* points in the plane, let  $G = (R \cup B, E)$  be the complete weighted bipartite graph with the property that there exists an edge between two points if and only if they are of different colors. Moreover, for each  $u \in R$  and  $v \in B$ , the weight  $d(u, v)$  of the edge  $(u, v)$  is given by the euclidean distance between the points  $u$  and  $v$ . The objective is to find a perfect matching in this graph whose cost is minimum.

The Euclidean Non-bipartite Matching Problem (ENM) can be defined similarly. Let  $V$  be a set of  $2n$  points in the plane. Consider the complete weighted graph  $G = (V, E)$  where the weight  $d(u, v)$  of an edge is given by the euclidean distance between the points  $u$  and  $v$ . Compute a perfect matching in this graph with minimum cost.

---

\*Department of Computer Science, Duke University

## 2 Exact Algorithms

### 2.1 Bipartite Case

Vaidya [4] gave an  $O(n^{5/2} \log n)$  algorithm for the EBM problem. Later Agarwal et. al. [6] improved the running time to  $O(n^{2+\epsilon})$ . Both the papers utilize the standard Hungarian method [8] for computing min-cost perfect matching in general bipartite graphs. The bipartite matching problem can be formulated as a linear program as follows. Associate a boolean variable  $x_{ij}$  with every edge  $(r_i, b_j)$ ,  $r_i \in R$ ,  $b_j \in B$ .  $x_{ij} = 1$  if the edge  $(r_i, b_j)$  belongs to the matching and  $x_{ij} = 0$  otherwise. The goal is to minimize  $\sum_{i,j} d(r_i, b_j)x_{ij}$  (the cost of the matching) subject to the constraints

- 1)  $\sum_j x_{ij} = 1$  for  $1 \leq i \leq n$
- 2)  $\sum_i x_{ij} = 1$  for  $1 \leq j \leq n$
- 3)  $x_{ij} \geq 0$  for  $1 \leq i, j \leq n$

It can be shown that an optimal solution to the above linear program will always assign integral values to the variables  $x_{ij}$ ,  $1 \leq i, j \leq n$ . The dual linear program can be written as

maximize  $\sum_i \alpha_i + \sum_j \beta_j$  subject to  $\alpha_i + \beta_j \leq d(r_i, b_j)$  for  $1 \leq i, j \leq n$ .

Here  $\alpha_i$  ( $\beta_j$ ) is the dual variable associated with the point  $r_i$  ( $b_j$ ). An edge  $(r_i, b_j)$  is *admissible* if  $\alpha_i + \beta_j = d(r_i, b_j)$ . A perfect matching  $M$  has minimum cost if there exists an assignment for the values  $\alpha_i$  and  $\beta_j$  satisfying the above constraints (namely,  $\alpha_i + \beta_j \leq d(r_i, b_j)$ ) and all the edges belonging to  $M$  are admissible [8]. A point is *exposed* if no edge in the matching is incident to it. A path is *alternating* if it alternately traverses edges of the matching. An *augmenting* path is an alternating path with the added property that both its start and end points are exposed. Starting with an empty matching  $M = \emptyset$ , the algorithm runs in  $n$  phases. In each phase, it finds an augmenting path  $\Pi$  all of whose edges are admissible. Every edge in  $\Pi$  belonging to  $M$  is discarded from  $M$ , and every edge in  $\Pi$  not belonging to  $M$  is included into  $M$ . Thus, the number of edges in  $M$  increases by 1 during each phase. For a general graph, every phase needs  $O(n^2)$  time. So the running time of Hungarian Method is  $O(n^3)$ . Although  $\Omega(n^2)$  space is required to represent a general complete weighted bipartite graph of  $2n$  vertices, an instance of the EBM problem requires only  $O(n)$  space (since the co-ordinates of the red and blue points uniquely determine the edge weights). Vaidya [4] showed that the underlying geometry of the EBM problem can be exploited to implement each phase in  $O(n^{1.5} \log n)$  time. One has to solve the *weighted closest pair* problem efficiently to speed up the run time of each phase: Given two sets of points  $P$  and  $Q$  in the plane, where  $w(u)$  denotes the *weight* associated with point  $u$  and  $d(u, v)$  denotes the euclidean distance between points  $u$  and  $v$ ; build a dynamic data structure for maintaining a pair of points  $(p, q)$ ,  $p \in P, q \in Q$  that minimizes the value of the expression  $d(p, q) - w(p) - w(q)$ . Agarwal et. al. [6] provided such a data structure with  $O(n^\epsilon)$  time per insertion or deletion, resulting in an improved running time of  $O(n^{1+\epsilon})$  for each phase.

## 2.2 Non-bipartite Case

Edmonds' algorithm [8] is a standard method for computing a min-cost perfect matching in general non-bipartite graphs. Vaidya [4] showed that geometry can be exploited to implement Edmonds' algorithm in  $O(n^{5/2} \log^4 n)$  time to solve the ENM problem. We give a brief overview of an  $O(n^{3/2} \log^5 n)$  algorithm proposed by Varadarajan [1]. A subset of points  $Q \subseteq V$  is called an *odd subset* if  $|Q| \geq 3$  and  $|Q|$  is odd. Associate a dual variable  $w_v$  with each point  $v \in V$  and a dual variable  $w_Q$  with each odd subset  $Q$ . For any point  $v \in V$ , let  $\lambda(v) = w_v + \sum_{v \in Q} w_Q$ . For every unordered pair of points  $u, v \in V$ , let  $\pi_{uv} = w_u + w_v + \sum_{|Q \cap \{u,v\}|=1} w_Q$ . A perfect matching  $M$  has minimum cost if each dual variable can be assigned a value in such a way that the following conditions hold true:

FEASIBILITY	$\pi_{uv} \leq d(u, v)$ for all $u, v \in V$
POSITIVE-DUAL	$w_Q \geq 0$ for every odd subset $Q$
ADMISSIBILITY	$(u, v) \in M$ only if $\pi_{uv} = d(u, v)$
MAXIMALITY	$w_Q > 0$ only if $M$ is maximal within $Q$

Certain odd subsets of  $V$  are designated as *blossoms*. Any two distinct blossoms are either mutually exclusive, or one of them is the superset of the other. A blossom is *outermost* if it is a subset of no other blossom. The outermost blossoms partition the set  $V$  into mutually exclusive subsets. Each blossom has exactly one point (called the *base*) that is not matched to any other point in the same blossom. An outermost blossom is exposed if its base is not incident to any edge of the matching. The algorithm maintains the following invariant: if an odd subset  $Q$  is not a blossom, then  $w_Q = 0$ . Given a set of points  $P$  in the plane, a *separating circle* of  $P$  has the following properties.

- 1) There are at least  $|P|/4$  points in  $P$  lying inside (outside) the circle  $C$ .
- 2) Let  $\beta(p)$  denote the distance of a point  $p \in P$  from the circle  $C$ . Consider any subset  $P' \subseteq P$  of points such that for all distinct pair of points  $p, q \in P'$ ,  $d(p, q) \geq \beta(p) + \beta(q)$ . Then  $|P'|$  is at most  $O(\sqrt{|P|})$ .

The paper [1] showed that a separating circle for  $n$  points in the plane can be computed in linear time. The main algorithm employs the technique of geometric divide and conquer. It is a recursive procedure that takes as input a set of points  $U$  and a real number  $l_U(u)$  for each point  $u \in U$ . In the first call to the procedure,  $U$  is the given set of points  $V$  and  $l_V(v) = \infty$  for all  $v \in V$ . The procedure returns a matching of  $U$ , a set of blossoms in  $U$ , a set of dual variables  $w_u$  for each point  $u$  in  $U$ , and  $w_Q$  for each blossom  $Q$  in  $U$ . The dual variables have values assigned to them in such a way that the four constraints mentioned in the beginning of this section (namely, FEASIBILITY, POSITIVE-DUAL, ADMISSIBILITY and MAXIMALITY) are satisfied in addition to two more constraints.

- 1)  $\lambda(u) \leq l_U(u)$  for all points  $u \in U$ .
- 2) There exists some point  $q$  in every exposed blossom with the property that  $\lambda(q) = l_U(q)$ .

Given a set of points  $U$ , the procedure computes a separating circle  $C$  of  $U$ . Let  $U_1$  ( $U_2$ ) be the set of points in  $U$  lying inside (outside) the separating circle. The procedure calls itself recursively with the points set  $U_1$  and  $U_2$ . For every point  $u \in U_1$ ,  $l_{U_1}(u) = \min(l_U(u), \beta(u))$ . Similarly for every point  $u \in U_2$ ,  $l_{U_2}(u) = \min(l_U(u), \beta(u))$ . If we simply take the union of the returned matchings, blossoms and dual variables for the sets  $U_1$  and  $U_2$ , every constraint apart from the last one (which says every exposed blossom should have some point  $q$  with  $\lambda(q) = l_U(q)$ ) is satisfied. The conquer stage deals with this issue. Varadarajan [1] showed that the conquer stage can be implemented in  $O(n^{1/2})$  phases, each phase taking  $O(n \log^5 n)$  time. Since each of the subproblems has a size of at least  $n/4$ , we get the required running time of  $O(n^{3/2} \log^5 n)$ .

### 3 Approximation Algorithms

The first approximation algorithm for the ENM problem was given by Vaidya [5]. He proposed an algorithm that runs roughly in  $O(n^{3/2}/\epsilon^3)$  time and returns a matching whose cost is at most  $(1 + \epsilon)$  times the optimal. He also gave a heuristic with a running time of  $O(n \log^3 n)$  that guarantees an approximation ratio of  $3 \log_3(1.5n)$ . Arora [9] presented a Monte-Carlo  $(1 + \epsilon)$  approximation algorithm with time complexity  $O(n \log^{O(1/\epsilon)} n)$ . Employing the method of Arora [9] together with the divide and conquer scheme of Varadarajan [1], Agarwal et. al. [2] was able to improve the run time to  $O((n/\epsilon^3) \log^6 n)$ . Moreover, Rao et. al. [10] gave a constant factor Monte-Carlo approximation algorithm with time complexity  $O(n \log n)$  whose probability of success is at least  $1/2$ .

For the EBM problem, the paper [2] showed how to compute a  $(1 + \epsilon)$  approximation deterministically in  $O((n/\epsilon)^{3/2} \log^5 n)$  time. Now we briefly describe the article by Agarwal et. al. [3], which gave an  $O(\log(1/\epsilon))$  approximation algorithm with an expected running time of  $O(n^{1+\epsilon})$ . The paper utilizes the idea of randomly shifted quadtree. The algorithm is designed in such a way that the number of levels in the quadtree is  $O(\log(1/\epsilon))$ . Deviating from standard algorithms, a cell of a quadtree is sometimes partitioned into more than 4 subcells. The authors first show how to compute a number  $\alpha$  in  $O(n \log n)$  time with the property that the cost of the optimal matching lies in the range  $[\alpha, 2n^2\alpha]$ . It is assumed that all the points are enclosed in a bounding square with side length  $L$ . The crux of the algorithm is a recursive procedure which can be roughly described as follows. Depending on the values of  $\alpha, \epsilon, L$  and  $n$ , decide whether the input is *small* enough. If the input is small, solve it by either the Hungarian method or return some arbitrary perfect matching. Otherwise take a randomly shifted grid whose side length is chosen in some well defined manner. For each cell  $C$  of the grid, let  $R_C$  ( $B_C$ ) denote the number of red (blue) points contained in  $C$ . Let  $X_C = |R_C - B_C|$ . If the number of red (blue) points in  $C$  is greater than the number of blue (red) points, arbitrarily pick  $X_C$  red (blue) points and move them to the center of the cell  $C$ . After all the cells have been processed in this way, compute a perfect matching for the moved points using an algorithm for the transportation problem. For each cell  $C$ , recursively call the same procedure for computing a matching for the

unmoved points in  $C$ .

## 4 A Special Case

We conclude this article by describing how the Euclidean matching problem can be solved more efficiently if there is some restriction on the placement of input points.

Marcotte et. al. [7] presented  $O(n \log n)$  algorithms for both the EBM and ENM problems when the input points lie on the boundary of a convex polygon. We briefly describe the algorithm for the non-bipartite case. Let the points  $V = \{z_0, z_1, \dots, z_{2n-1}\}$  be ordered counterclockwise on the boundary of a convex polygon. Because of triangle inequality, no two edges of the optimal matching can cross each other. Define a set of edges as *extensible* if it is a subset of some optimal matching. The authors made the following observations.

- 1) Let  $\{z_l, z_{l+1}, \dots, z_k\}$  be any contiguous sequence of even number of points with an optimal matching  $M$ . If  $M$  contains the edge  $(z_l, z_k)$ , then the set  $M - \{(z_l, z_k)\}$  is extensible.
- 2) Let  $M$  be an extensible set of edges for point set  $V$  and let  $V'$  be the set of points adjacent to the edges in  $M$ . Now, if  $M'$  is an extensible set of edges for the point set  $V - V'$ , then  $M \cup M'$  is extensible for point set  $V$ .

For the sake of notational convenience, relabel the input sequence of points  $z_0, z_1, \dots, z_{2n-1}$  as  $x_0, y_0, x_1, y_1, \dots, x_{n-1}, y_{n-1}$ . The algorithm follows divide and conquer paradigm. It partitions the input set of points into two equal sized subsets  $V_1 = \{x_0, y_0, \dots, x_{n/2}, y_{n/2}\}$  and  $V_2 = \{x_{n/2+1}, y_{n/2+1}, \dots, x_{n-1}, y_{n-1}\}$ . The optimal matchings of  $V_1$  and  $V_2$  (denoted by  $M_1$  and  $M_2$ ) are computed recursively. The union of  $M_1, M_2$  and the boundary edges of the polygon constitutes an outerplanar graph. Let  $M_{12}$  denote the set of edges in  $M_1 \cup M_2$  not lying in the same face as that of  $(x_0, y_{n-1})$ . Observations 1 and 2 imply that the set  $M_{12}$  is extensible. Let  $A = \{x_0, y_0, \dots, x_{m-1}, y_{m-1}\}$  ( $B = \{x_m, y_m, \dots, x_{r-1}, y_{r-1}\}$ ) be the endpoints of the edges in  $M_1 - M_{12}$  ( $M_2 - M_{12}$ ) sorted in counterclockwise order on the boundary of the polygon. Since the edges in an optimal matching are non-crossing, we only need to consider edges of the form  $(x_i, y_j)$  while constructing an optimal matching. Let there be a weight function  $w$  mapping each point in  $A \cup B$  to some real number such that  $w(x_0) = 0$  and for any two consecutive points  $p, q$  in the sequence  $\{x_0, y_0, \dots, x_{r-1}, y_{r-1}\}$ ,  $w(p) + w(q) = d(p, q)$ . For any pair of points of the form  $(x_i, y_j)$ , where either  $x_i \in A, y_j \in B$  or  $x_i \in B, y_j \in A$ , let  $D(x_i, y_j) = d(x_i, y_j) - w(x_i) - w(y_j)$  denote the *weighted distance* between them. Marcotte et. al. [7] proved that if every  $D(x_i, y_j)$  other than  $D(x_0, y_{r-1})$  is non-negative, then either the matching  $\{(x_0, y_0), \dots, (x_{n-1}, y_{n-1})\}$  or the matching  $\{(y_0, x_1), \dots, (y_{r-1}, x_0)\}$  is optimal for  $A \cup B$ . Otherwise the algorithm finds a *critical* edge  $(x_k, y_l)$  which leads to an easy identification of a further set of extensible edges. Thus, the size of the problem is reduced. The problem is similar to finding the weighted closest pair as defined in Section 2.

Although the above algorithm is for a convex polygon, it can be easily extended to

cases where the polygon is non-convex. The running time for the non-convex case is  $O(n \log^2 n)$ . The bipartite version of the matching problem on a convex polygon can be solved in  $O(n \log n)$  time using similar methods. The article [7] also considered the following verification problem: Given a matching  $M$ , decide whether or not it has minimum weight? It turns out that this problem can be solved in  $O(n\alpha(n))$  time for a convex polygon and in  $O(n\alpha(n) \log n)$  time for a non-convex polygon, where  $\alpha(n)$  is the inverse Ackermann function.

## References

- [1] K.R. Varadarajan. A Divide-and-Conquer Algorithm for Min-Cost Perfect Matching in the Plane. FOCS (1998).
- [2] P.K. Agarwal and K.R. Varadarajan. Approximation Algorithms for Bipartite and Non-bipartite Matching in the Plane. SODA (1999).
- [3] P.K. Agarwal and K.R. Varadarajan. A Near-Linear Constant-Factor Approximation for Euclidean Bipartite Matching? SoCG (2004).
- [4] P.M. Vaidya. Geometry Helps in Matching. SIAM J. Computing, 18: 1201-1225 (1989).
- [5] P.M. Vaidya. Approximate Minimum Weight Matching on Points in  $k$ -Dimensional Space. Algorithmica, 4: 569-583 (1989).
- [6] P.K. Agarwal, A. Efrat and M. Sharir. Vertical Decomposition of Shallow Levels in 3-Dimensional Arrangements and Its Applications. SoCG (1995)
- [7] O. Marcotte and S. Suri. Fast Matching Algorithms for Points on a Polygon. FOCS (1989)
- [8] E. Lawler. Combinatorial Optimization: Networks and Matroids. Holt Rinehart and Winston, New York (1976).
- [9] S. Arora. Nearly Linear Time Approximation Schemes for Euclidean TSP and Other Geometric Problems. FOCS (1997)
- [10] S.B. Rao and W.D. Smith. Improved Approximation Schemes for Traveling Salesman Tours. STOC (1998)